Computational Visual Media
https://doi.org/10.1007/s41095-019-0141-4 Vol. 5, No. 2, June 2019, 161–170
Research Article

Seamless and non-repetitive 4D texture variation synthesis and real-time rendering for measured optical material behavior

Martin Ritz[1] (✉), Simon Breitfelder[2], Pedro Santos[1], Arjan Kuijper[1,2], and Dieter W. Fellner[1,2,3]

Abstract We show how to overcome the single weakness of an existing fully automatic system for acquisition of spatially varying optical material behavior of real object surfaces. While the expression of spatially varying material behavior with spherical dependence on incoming light as a 4D texture (an ABTF material model) allows flexible mapping onto arbitrary 3D geometry, with photo-realistic rendering and interaction in real time, this very method of texture-like representation exposes it to common problems of texturing, striking in two disadvantages. Firstly, non-seamless textures create visible artifacts at boundaries. Secondly, even a perfectly seamless texture causes repetition artifacts due to their organised placement in large numbers over a 3D surface. We have solved both problems through our novel texture synthesis method that generates a set of seamless texture variations randomly distributed over the surface at shading time. When compared to regular 2D textures, the inter-dimensional coherence of the 4D ABTF material model poses entirely new challenges to texture synthesis, which includes maintaining the consistency of material behavior throughout the 4D space spanned by the spatial image domain and the angular illumination hemisphere. In addition, we tackle the increased memory consumption caused by the numerous variations through a fitting scheme specifically designed to reconstruct the most prominent effects captured in the material model.

1 Fraunhofer IGD, Darmstadt, 64283, Germany. E-mail: M. Ritz, martin.ritz@igd.fraunhofer.de (✉); P. Santos, pedro.santos@igd.fraunhofer.de; A. Kuijper, arjan.kuijper@igd.fraunhofer.de; D. Fellner, dieter.fellner@igd.fraunhofer.de.
2 TU Darmstadt, Darmstadt, 64289, Germany. E-mail: simon.breitfelder@gmail.com.
3 TU Graz, Graz, 8010, Austria.
Manuscript received: 2019-03-20; accepted: 2019-03-29

Keywords optical material behavior; reflectance modeling; 4D texture synthesis; texturing

1 Introduction and related work

3D rendering research is continuously striving to come closer to a physically realistic representation of real-world surfaces, as it is ever more widely applied to fields where a high degree of realism is required (e.g., the 3D games industry) or where rapid prototyping is applied in early stages of design (e.g., in the automotive and textile industries). The approach of measuring ABTFs (approximate bi-directional texturing functions) [1, 2] is one way to avoid the need for synthetic design of material models, as it used to be the standard for a long time even though requiring significant manual effort. ABTFs were first proposed by Ref. [1], who used a quarter light arc with incident illumination from point lights, mounted at angles ranging from zero elevation (grazing angles onto the material surface) up to nearly vertically incident light directions. The acquisition of optical material behavior, while already spatially varying due to the matrix sensor of the camera, relied on the assumption of isotropy, meaning that the material sample response is not affected by its rotation around its surface normal. The resulting ABTF data consisted of two spatial dimensions (within the sample surface) and one dimension for light variation, leading to three dimensions in total. Many materials cannot be faithfully represented by assuming isotropy, as they are a combination of many different materials each with different physical structure, and thus reacting differently at each individual surface point to changes of the incoming light direction around the surface normal. Thus, an additional dimension of incoming light direction was added in Ref. [2]. By using a

turntable which rotates the sample around its surface normal under the camera, 4D ABTF model results from combining the two lateral dimensions with two angular dimensions describing the hemisphere of incident light directions. Also, the acquisition process was entirely automated and reduced to only a few minutes per sample. Effectively, the combinatorial use of rotary and quarter light arc leads to an illumination hemisphere with the camera at its center, looking vertically down on the sample. The ABTF material model acquired and rendered with the technique in Ref. [2] represents actually captured real-world material behavior as a 4D texture that can be rendered in real time. By only considering one perspective vertically above the sample, it provides an abstraction of the higher dimensional 6D BTF (bi-directional texturing function) [3] that captures the spatially varying material behavior of flat materials, discretized to the resolution of a matrix sensor (2D), for all combinations of incoming light (2D) and outgoing observer direction (2D), while preserving the dependence on the incoming light direction.

Figure 1 puts the ABTF model within the context of the taxonomy of surface reflectance functions. It is represented by a two-dimensional stack of images (layers), each of which expresses the spatially varying material behavior on the sample surface for one specific incoming light direction. Rendering is done by the shader accessing the captured image stack, according to the angle of the virtual light source in the scene in relation to the surface normal and the current U–V coordinates specified to the shader during rendering of the current fragment, provided by the texture mapping of the 3D geometry to be rendered. The texture-based nature of the ABTF model allows for arbitrary mapping onto any 3D geometry with free choice of scaling and orientation, and the abstraction from multiple observer directions allows the model to fit into current graphics memory for real-time rendering.

Despite this abstraction, the material model provides compelling results for most materials, while shortcomings only arise where material behavior actually depends on the observer direction, for instance for materials with a "chameleon" effect that change colors depending on the viewing angle. However, the texture-like nature of the model shares common texturing problems, which becomes apparent on two levels. Firstly, if single layers in the 4D texture stack are not synthesized to be seamless, border artifacts become apparent in the form of spatial discontinuities along the seams across texture units. Secondly, even a perfectly synthesized seamless texture causes repetition artifacts that strike the eye when large numbers of identical patches are placed side-by-side over the 3D surface once they are visible within the field of view at the same time. These two unsolved problems are the ones addressed by

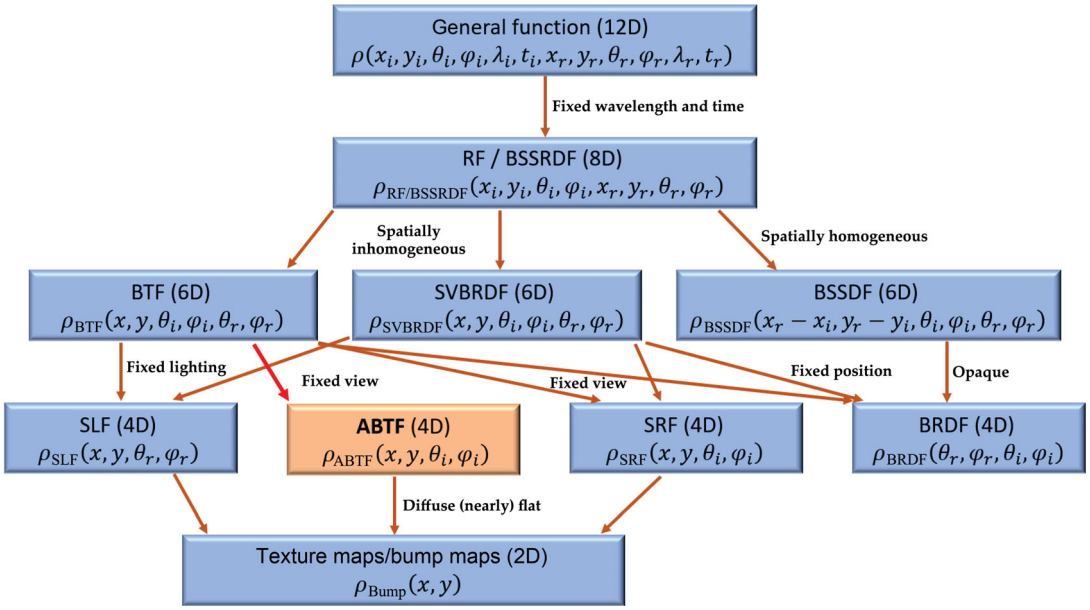


Fig. 1 ABTF model in the surface reflectance function taxonomy introduced in Ref. [4], derived from BTFs and depending on the same variables as surface reflectance functions (SRF), but with different semantics.

this work.

Our solution to both problems is a novel texture synthesis approach that first makes the single ABTF texture layers seamless, and in a second step generates a set of variations for each layer. The patch variations are precomputed and randomly distributed on the surface at shading time, thus completely disrupting any regular patterns and entirely removing repetition artifacts, generating the impression of a non-repeating artifact-free surface (see Fig. 2).

For the domain of 2D texturing, there are approaches that solve these challenges [5, 6]. However, we are dealing with inter-dimensional coherence within a 4D texture, posing the strong requirement of maintaining consistency across the multi-dimensional space of measured material behavior information, represented by the spatial domain and the two dimensions defined by a hemisphere of incoming light directions. This introduces challenges such as specular highlight responses for which continuity must be ensured during realignment of texture sub-regions. Finally, generating multiple variations per ABTF layer increases memory consumption. We counter this problem using a fitting scheme specifically designed to reconstruct the most prominent effects captured by the material model.

2 Technical approach

2.1 Texture synthesis and periodization

The synthesis method for measured ABTF materials developed in this work is based on image quilting [7], which assembles a new texture by transferring small patches from an input image to different new locations within the newly synthesized texture while maintaining optical similarity at the transitions within overlapping regions of neighboring patches. We extended it by adding initial reassembly phases before the original process, ensuring periodicity at the boundary regions of the texture to be synthesized so that it can be seamlessly concatenated border to border in both dimensions. Most importantly, we use the output of the algorithm not just to generate a new patch placement distribution for one texture, but instead for a non-deterministically generated set of patch transfer prescriptions consistently applied to all ABTF data set layers, which guarantees consistency across different incident light angles and provides a random patch placement every time to avoid repetition artifacts.

The algorithm starts with an empty destination texture matching the input image dimensions. It consists of four phases, visualized in Fig. 3.

- **Phase 1**: Patches are placed at all corners to ensure periodicity, which is achieved by choosing a random source patch and splitting it vertically and horizontally through its center for transfer to diagonally opposing corners of the target texture. As the inner patch split edges coincide with the respective outer texture edges, appending the target texture at any edge reunites the original patch, thus leading to artifact-free periodicity for the corner pieces.
- **Phase 2**: A similar idea is followed to fill the horizontal edges of the target texture, with two differences. Firstly, the source patch to be

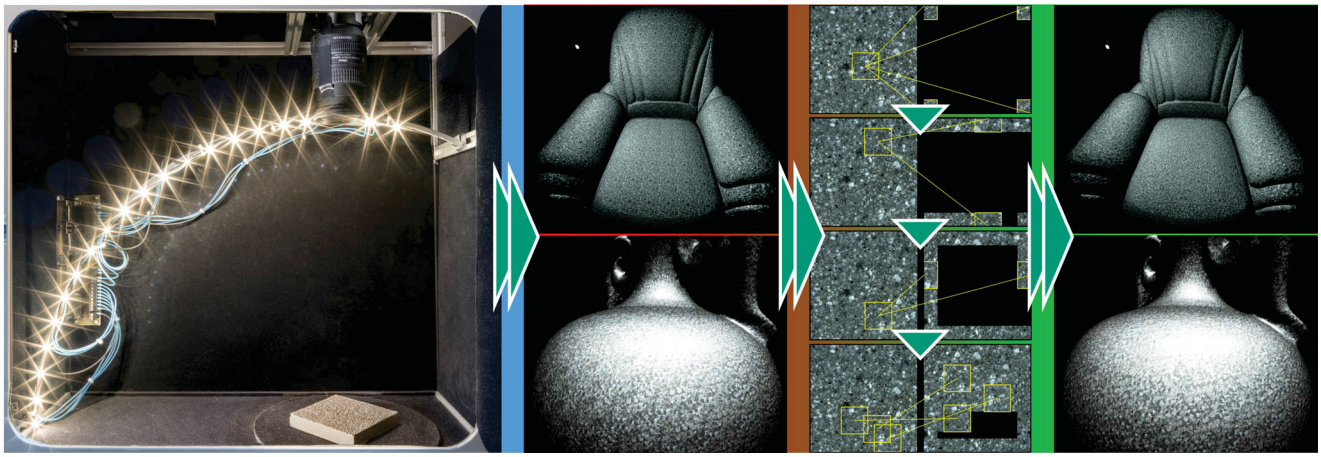


Fig. 2 From measured optical material behavior (4D textures with seams and repetitions) via synthesis of a set of seamless and non-repetitive 4D texture variations to real-time 3D rendering on arbitrary 3D geometry, without texturing artifacts.

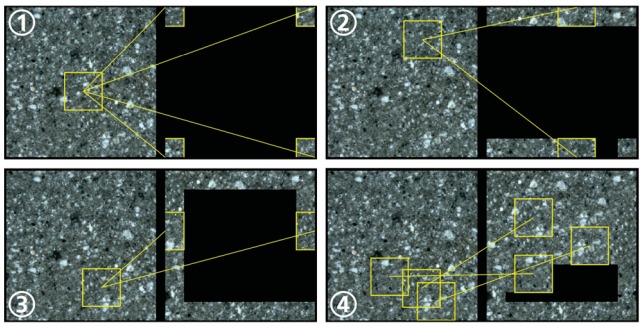


Fig. 3 Visualization of phases 1–4 of the algorithm. Left: source image. Right: synthesized texture.

transferred is now horizontally split into two halves which then are transferred to vertically opposing edges of the target texture, ensuring periodicity as in Phase 1. Secondly, to avoid deterministic behavior, the patch to be transferred is now chosen at random from a set of best matching patches. Best matches are determined based on the optical similarity between a patch and the existing target neighborhood measured in their common region of overlap, as explained in detail in Section 2.3.

- **Phase 3**: Steps analogous to Phase 2 are performed to fill the vertical edges while ensuring periodicity.
- **Phase 4**: The inner area of the target texture is filled following a row-major traversal scheme. A set of best matching source patches is determined for each empty block, of which one is chosen for transfer at random. Again, as in Phases 2 and 3, optical similarity is used as a metric to establish the most continuous spatial transition across seams within the target patch neighborhood.

Parameters for this algorithm are: the set size for best matches, the block size of texture patches to be transferred, and the degree of overlap between a patch candidate and its existing destination neighborhood.

2.2 Visual similarity metric

The metric used to determine optical similarity to guide the process of finding best matching patches for transfer (see Section 2.3) is a monochrome error image ΔI (see Fig. 5(1), gray scale region), covering exactly the area of overlap currently considered. The intensity for each pixel is computed as the sum of squared differences between source and destination image space for the three color channels evaluated at that pixel position. For 2D textures, the texture is the source image space for this computation, while for an ABTF dataset, consistency throughout the 4D texture must be maintained to avoid visual artifacts caused by inconsistencies between texture layers captured with different lighting directions, so a suitable image space must be chosen that represents the entire ABTF dataset.

In Ref. [8] a BTF dataset is represented by a height map computed from all images captured, representing the approximate geometry of the surface. In addition to the height map, we compute a diffuse map and a reflectance map (see Fig. 4). The three images represent the most relevant aspects of the entire ABTF dataset and are combined by weighted per-pixel summation into one reference image ΔR, serving as source image space. During patch transfer, the impact of the respective diffuse, height, and reflectance component contributions is controlled by weights. Computing ΔI is the most computationally intensive step, as it has to be done for each overlapping region evaluated during the search for best matches. The algorithm described in Section 2.1 has to run in the serial order described, as subsequent patch placement is dependent on previously placed patches. The sole purpose of the reference image R is to serve as basis for generation of a set of transfer prescriptions, which are then applied identically to each texture layer in the ABTF dataset.

The diffuse map is the normalized sum of all images captured for each lighting direction. The surface geometry is represented by a height map which is computed in two steps. Firstly, a normal map is estimated by evaluating the strongest response for each pixel over all lighting directions. The vectors computed here do not generally represent the surface normal, but the averaged direction to the strongest incoming light contribution. This difference only applies to strongly reflective materials, while there is no difference for perfectly diffuse materials. The height map follows in the second step from integration of the normal map [9]. The reflectance map of the

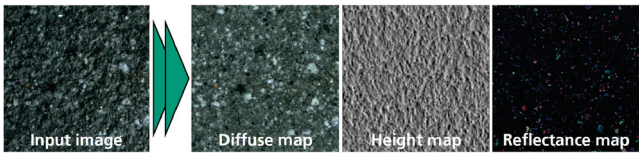


Fig. 4 Left: single example image from the ABTF dataset. Right: reference image components for patch transfer guidance based on optical error, derived from the entire ABTF dataset to represent surface structure and material behavior.

surface is computed by comparing two normal maps weighted with different exponents to find strongly reflective areas on the surface.

2.3 Seamless transfer

Naive copying of rectangular areas leads to highly noticeable edges at the boundaries even if the general structure fits the local neighborhood (see Figs. 5(1) and 5(2)). Image quilting reduces this effect by allowing a free-form boundary cut which minimizes optical differences between source patch and target environment (see Figs. 5(3) and 5(4)). Optical differences are expressed by the error image ΔI introduced in Section 2.2, which at the same time works as a mask, excluding all pixels set to zero (black) from the error metric (see Fig. 5(1)).

Finding the free-form boundary cut following the path with the minimum cumulative error in ΔI is a path-finding problem which can be solved using Dijkstra's algorithm. The algorithm for finding the free-form boundary cut presented here computes many path candidates in a set of sub-regions of ΔI. Every pixel in a sub-region represents a node in the Dijkstra graph. Interconnections are generated for all adjacent pixels (only horizontally and vertically, not diagonally) using the average error (i.e., intensity value in ΔI) between both pixels as cost. The inner region is masked out (see the black square in the center of Figs. 5(1), 5(3), and 6) as this part of the patch is adopted unchanged and must thus not be crossed by the boundary cut. At first, only the sub-regions at the left, top, right, and bottom edges are considered. All combinations of start and end points on the yellow lines for each of the four highlighted regions are evaluated using Dijkstra's algorithm, leading to the path with minimum cost for the next step (see Figs. 6(2) and 6(3)). The endpoints of the selected paths (yellow in Fig. 6(4)) are then connected through the corner-regions which requires only one evaluation of Dijkstra's algorithm per corner. The full cyclic path (see Fig. 6(5)) represents the minimum error boundary cut.

Fig. 5 Optical error image ΔI (1); naive transfer of rectangular patch exposing visual discontinuities (2); minimum boundary cut through optical error field (3); transfer of boundary region (4).

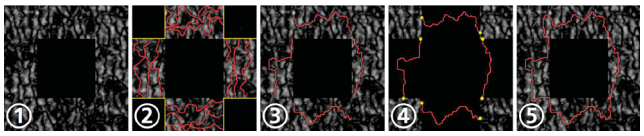


Fig. 6 Optical error image ΔI (1); possible minimal paths through the four edge-regions (2); paths with minimal cost (3); the four path segments (endpoints yellow) are connected (4); resulting cyclic boundary cut (5).

To improve the resulting image quality, the boundary cut is not used as a hard edge between patch content and existing destination neighborhood. Instead, blur masks are created for the areas inside and outside the boundary cut using Gaussian blurring (see Fig. 7). The patch content and the neighborhood background are then combined by weighting their respective masks, leading to a smooth transition.

As the resulting texture is generated iteratively, some regions do not yet contain data and thus do not provide a basis for error computation, e.g., the black region in the example in Fig. 8(left). When placing a patch at the right border region the overlap error can be computed only for the region of overlap in the center of Fig. 8(middle, shown as grayscale error image). As the error for all paths through yet undefined (empty) regions is 0, the decision for the best path is ambiguous and can yield bad results. The red line in Fig. 8(middle, above) represents the path with minimal error, but does not consider the entire viable area within the existing neighborhood, resulting in visible discontinuity artifacts highlighted by the red rings in the resulting texture in Fig. 8(right, above). We solve this problem by introducing border-flags for the main directions left, top, right, and bottom, that force the path-finding algorithm to enclose the full region inside the boundary cut towards the directions indicated by the respective flags. The boundary cut in Fig. 8(middle, below) results from setting the top and bottom border-flags, leading to a soft transition between both patches as during computation of the optimal path, the entire border region is considered as visible in Fig. 8(right, below).

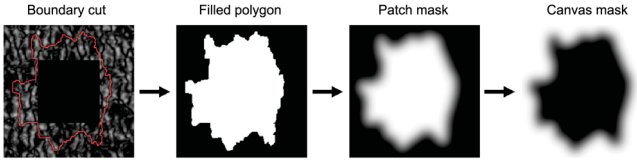


Fig. 7 Boundary cut used to draw a filled polygon defining the weight of the patch content to smooth the transition using Gaussian blur.

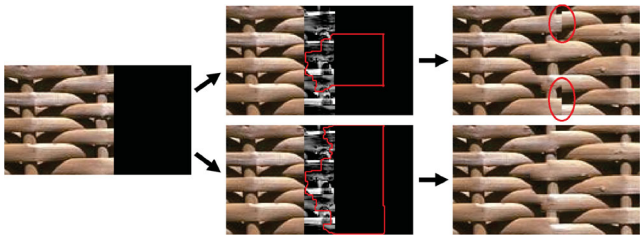


Fig. 8 Border flags controlling the extent of the region considered by visual matching during patch placement. Left: initial coverage with undefined regions on the right. Middle: error image in overlap region (grayscale) and two possible candidates for the boundary cut assigned the same cost in the yet empty areas of the canvas. The upper candidate does not consider the neighboring regions above and below the existing patch, while the lower candidate is aware of the unset regions. Right: difference in results.

2.4 Variation synthesis and rendering

Periodic regular textures can be mapped to large surfaces back to back without visible repetition artifacts, but in the case of irregular and stochastic textures, highly noticeable periodic patterns as in Fig. 9(left) appear. This problem can be handled by generating tile sets, within which certain combinations of elements are periodic, and the variation of different combinations can be used to fill a regular grid and thus create a textured surface without visible seams and repetitions. Many tile sets are based on Wang tiles [5] that define different types of edges and compute a tile for every combination of edge types so that there is always a compatible tile for a specific neighborhood of adjacent edges. Only tiles with the same type (types visualized in Fig. 10 in different colors) can be placed next to each other to share one edge.

The variation synthesis technique introduced in this work generates a corner-based tiling similar to colored corners tiling [10]. A complete tile set created using V corner types yields V^4 tiles in total. Already using $V = 2$ leads to 16 tiles which removes repetition artifacts, as the comparison in Fig. 9 shows. Figure 11

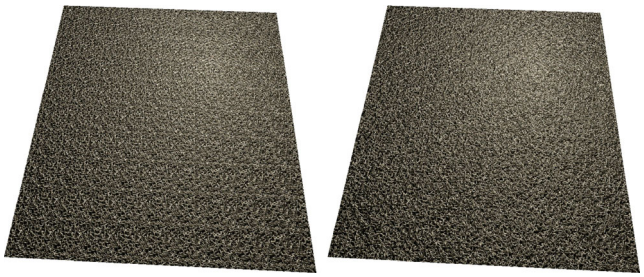


Fig. 9 Left: periodic pattern caused by repeating the same texture 10 times in both dimensions. Right: the same tiling rendered with randomly distributed seamless texture variations.

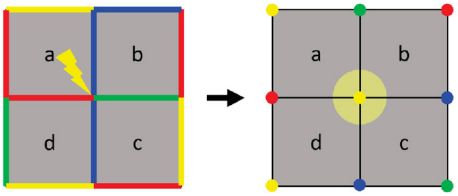


Fig. 10 Tiles incompatible at common corner despite compatible edges, solved by enforcing common corners.

gives a schematic overview of the corner compatibility within a set of 16 tile variations, where compatibility between neighboring tiles is expressed by matching color. Compatibility between neighboring tiles is strongly limited by edge compatibility and is only given for a subset of the tile set.

A pseudo-random number generator implemented in the shader as a hash function guarantees random selection of variations for mapping on the surface during rendering while keeping the overall mapping on the entire object surface constant without the need to store the chosen assignment of variations. Accessing texture patches at rendering time requires consistency, since texture variations once chosen and mapped to a rendered part of the object must be chosen identically each subsequent time the respective area is rendered. We use a pseudo-random number generator to satisfy this requirement. The generator works similar to a hash function that serves as deterministic, surjective function, assigning always the same function values to the same parameters, but potentially also assigning the same function value to different parameters. This is acceptable as the number of texture variations is limited and has to be reused multiple times across the object surface. The generator accepts an initial seed value that can be used to control the texture variation distribution and easily generate different renderings

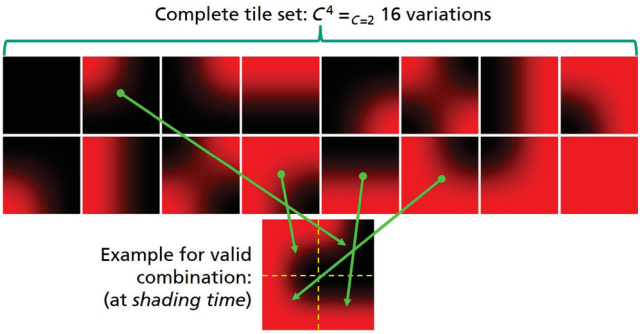


Fig. 11 Abstract tile set of minimum size based on 2 corner types (red and black), leading to 16 variations, indicating corner compatibilities (above). Example of 2×2 neighborhood construction (below).

using the same input texture and 3D model. The available texture patch variations are addressed using U–V mapping, where multiples of the default range $[0, 1]$ allow addressing of the virtual array of texture variations (see Fig. 12).

2.5 Fitting

The smallest possible tile set using $V = 2$ corner types ($V = 1$ is equivalent to a single periodic texture) leads to $2^4 = 16$ texture variations in total (see Fig. 11). The total memory consumption depends on the image resolution $W \times H$ and the sampling density of the virtual lighting hemisphere, expressed by R rotations of the sample and E discrete elevations of the light source, and amounts to $M = 3W \times H \times R \times E$ bytes. The data sets used in this work range from 300 MB to 1 GB, which leads to a total memory consumption of 4.7–16 GB including all texture variations. This amount exceeds typically available graphics memory and also would result in poor frame rates because of the lack of memory locality, as the renderer needs to access a widespread range of data when the surface is rendered under varying illumination. The key to real-time processing of large tile sets is thus compression. We use a fitting scheme we designed explicitly for this challenge. We limit the memory consumption for $V = 2$ corner types, or 16 texture variations, per ABTF data set to under 2 GB by fitting an analytic model based on the HSL color space (Fig. 13(right)). The choice of this specific color space for fitting enables reconstruction of the reflectance behavior for each single pixel on the sample surface (Fig. 13(left)) without visual degradation.

Analysis of the reflectance function for different pixels (Fig. 14), transformed into HLS color space, confirms that hue remains mostly constant over the entire spectrum of incident illumination angles as can be seen in Fig. 15 for one specific pixel, evaluated in

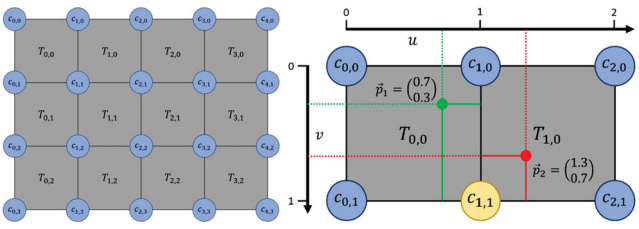


Fig. 12 Left: different texture variations $T_{u,v}$ side by side with shared corners $C_{i,j}$. Right: texture patches are addressed using U–V mapping. Different variations are accessible by using multiples of the default range $[0, 1]$.

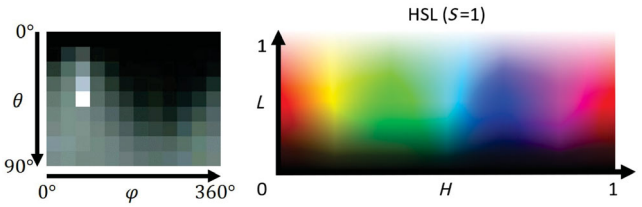


Fig. 13 Left: visualization of reflectance behavior for one pixel for full angular spectrum (light source elevation θ and rotation angle φ). Right: HSL color spectrum.

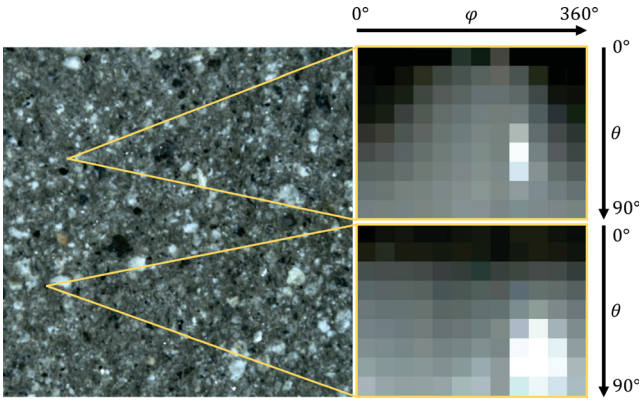


Fig. 14 Reflectance functions for single pixels. Left: image captured at $\varphi = 90°, \theta = 79°$. Right: corresponding reflectance functions for the two highlighted pixels.

HLS for the entire 2D illumination space. Similarly, saturation remains mostly unaffected for changing illumination directions. This observation allows us to set hue and saturation constant. The reason why we use the HLS model rather than HSV or similar color models is that the lightness value alone controls the transition from black via color to white, while in other models, this transition requires changing further parameters such as the saturation value.

The lightness value for the two-dimensional lighting direction (azimuth, elevation) is reconstructed using six parameters per azimuth light angle, plus two parameters for hue and saturation, for every pixel.

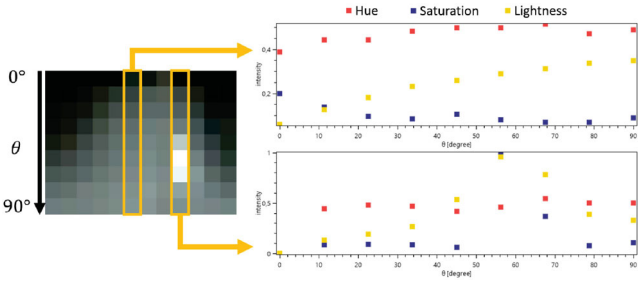


Fig. 15 Lightness curves of a pixel for different angles of rotation in HSL color space. Note how hue (red) and saturation (blue) remain almost constant. The deviation in the graph is due to the specular highlight at the respective position.

In compressed form the tile sets fit into typically available graphics memory and can thus be rendered in real time.

A visual comparison for two rendered 3D models mapped with two different physically measured ABTF material samples processed by this method is given in Fig. 16. The renderings on the left were generated based on the original uncompressed ABTF dataset (reference), shown side by side with a rendering using the same settings, but this time evaluating the compressed dataset after applying the fitting scheme described (final output of the algorithm shown). The only perceivable visual difference is a slight discrepancy in hue. The reason for this effect is a consequence of the assumption that for all variations of incoming light directions, the hue channel of the material response expressed in the HLS color model remains mostly constant, which is only approximately the case as can be seen in Fig. 15(red curve).

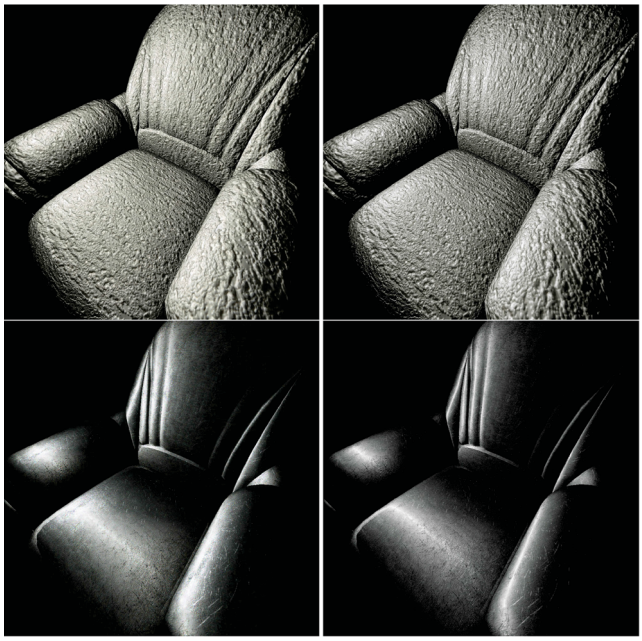


Fig. 16 Visual comparison of original (left) and compressed (right) ABTF datasets, captured from two different physical material samples "Ingrain" (above) and "Metal" (below). Both material samples feature a wide spectrum of optical effects like self shadowing and anisotropic reflections, which are successfully reconstructed by the fitting model presented in this work.

3 Results

The synthesis method for measured ABTF material models developed in this work is based on image quilting [7] which assembles a new texture by transferring small patches from an input image of identical dimensions to different new locations within the new image while maintaining optical similarity within overlapping regions of neighboring patches. We extended it by adding initial reassembly phases during which the target texture is made periodic at its boundary regions so that it can be seamlessly concatenated border to border in both dimensions. Most importantly, we use the output of the algorithm not just to generate a new patch placement distribution for one texture, but instead for a non-deterministically generated set of patch transfer prescriptions applied to all ABTF data set layers, which both guarantees consistency for different light angles and provides a random patch placement for every texture variation to avoid repetition artifacts. The consequence of using an entire set of texture variations for all different illumination angles is respective increase in memory consumption. Exploiting the fact that only one dimension is

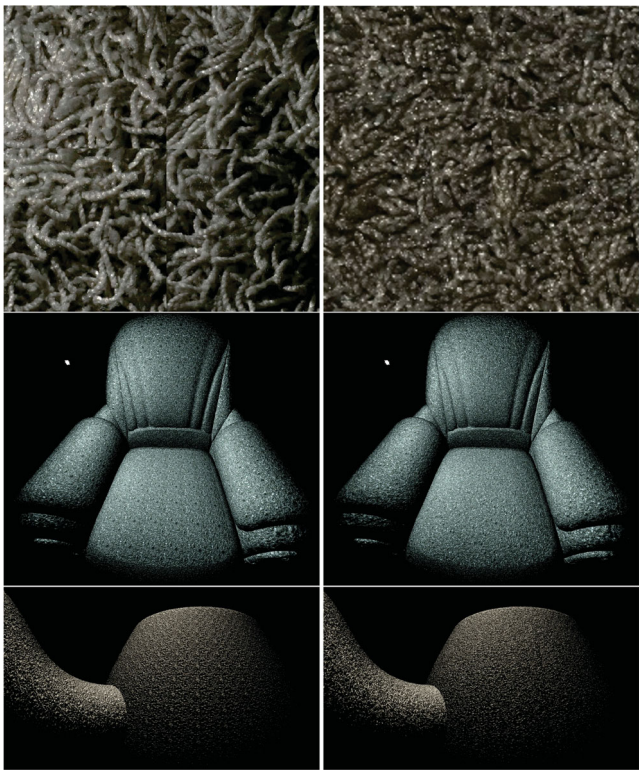


Fig. 17 3D rendering of the ABTF material model. Left: state of the art before this work. Right: our results. Above: plane mapped with 2×2-tiled texture, showing clear discontinuity boundaries at the seams, that are entirely removed by our solution. Center: armchair model mapped with multi-repetition stone texture. Our results do not show any repetition artifacts. Below: teapot model mapped with carpet texture, exposing repetitions of identical textures, while our result seems to consist of one non-repetitive texture only.

significantly affected in HLS color space by changes in the direction of incident illumination, we compensate for the increased memory consumption by a novel fitting scheme. Consequently, the reflectance behavior for each single pixel on the sample surface is reconstructed without visual degradation, while 3D rendering of models mapped with texture variation-based ABTF data sets can still be done in real time.

4 Conclusions and future work

We have removed the most significant limitation of an existing 4D texture-based system for acquiring spatially varying optical material behavior of real object surfaces, in that now neither texture seam artifacts nor repetition artifacts disturb the compelling visual experience when rendering the material applied to arbitrary 3D geometries, even if the textures are mapped to large surfaces with many repetitions.

Not only have we extended current work on 2D texture synthesis by a number of optimizations providing more robustness across the different classes of textures (regular, near-regular, irregular, and stochastic), but we have especially tackled entirely new challenges posed by the higher dimensionality and inter-dimensional correlation of 4D ABTF material texture sets, and compensated for the resulting increase in memory consumption such that 3D rendering can still be done in real time, just like with the previous approach, but without causing texture artifacts.

As future improvements, we see firstly that the quality of synthesized tile sets can be automatically adjusted according to the occurence of prominent patterns within the texture that draw the observer's attention, since periodicity artifacts depend strongly on the texture class. Secondly, to avoid noisy rendering for very dense tiling due to poor surface sampling, mipmaps adjusted to texture variations can be used that store a texture pyramid with increasing resolution to adaptively react to the sampling resolution.

Acknowledgements

This work was partially supported by the European project MAXIMUS (No. FP7-ICT-2007-1-217039) and the German Federal Ministry for Economic Affairs and Energy project CultLab3D (No. 01MT12022E).

References

[1] Kautz, J.; Sattler, M.; Sarlette, R.; Klein, R.; Seidel, H.-P. Decoupling BRDFs from surface mesostructures. In: Proceedings of Graphics Interface 2004, 177–182, 2004.

[2] Ritz, M.; Santos, P.; Fellner, D. Automated acquisition and real-time rendering of spatially varying optical material behavior. In: Proceedings of the ACM SIGGRAPH 2018 Posters, Article No. 33, 2018.

[3] Schwartz, C.; Sarlette, R.; Weinmann, M.; Klein, R. Dome II: A parallelized BTF acquisition system. In: Proceedings of the Workshop on Material Appearance Modeling, 2013.

[4] Weinmann, M.; Klein, R. Advances in geometry and reflectance acquisition (course notes). In: Proceedings of the SIGGRAPHAsia 2015 Courses, Article No. 1, 2015.

[5] Cohen, M. F.; Shade, J.; Hiller, S.; Deussen, O. Wang Tiles for image and texture generation. *ACM Transactions on Graphics* Vol. 22, No. 3, 287–294, 2003.

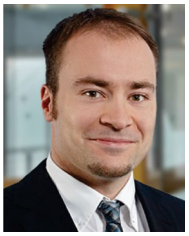
[6] Ng, T. Y.; Wen, C.; Tan, T. S.; Zhang, X.; Kim, Y. J. Generating an /spl omega/-tile set for texture synthesis. In: Proceedings of the International 2005 Computer Graphics, 177–184, 2005.

[7] Efros, A. A.; Freeman, W. T. Image quilting for texture synthesis and transfer. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 341–346, 2001.

[8] Liu, X.; Yu, Y.; Shum, H.-Y. Synthesizing bidirectional texture functions for real-world surfaces. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 97–106, 2001.

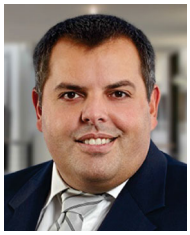
[9] Nozick, V.; Ismael, D.; Saito, H. GPU-based photometric reconstruction from screen light. In: Proceedings of the 8th Annual International Conference on Artificial Reality and Telexistence, 242–245, 2008.

[10] Lagae, A.; Dutré, P. An alternative for Wang tiles: Colored edges versus colored corners. *ACM Transactions on Graphics* Vol. 25, No. 4, 1442–1459, 2006.



Martin Ritz is deputy head of the Dept. of Cultural Heritage Digitization at Fraunhofer IGD. Next to technical coordination, his research targets acquisition of 3D geometry and optical material behavior, describing light interaction of objects for arbitrary combinations of light and observer directions. He received his bachelor and master of science degrees in informatics in 2006 and 2009 from TU Darmstadt, respectively, as well as his master of science degree in computer science in 2008 from the University of Colorado at Boulder.

Simon Breitfelder completed his bachelor and master degrees in computer science at TU Darmstadt in 2011 and 2018, respectively. He focused on computer graphics topics during his studies and is now working as software engineer at DENIC eG.



Pedro Santos is head of the Dept. of Cultural Heritage Digitization at Fraunhofer IGD. He received his diploma and master degrees from the University of Applied Sciences in Darmstadt, Germany and the Technical University of Lisbon, Portugal, respectively. His research interest is autonomous, color-calibrated 3D mass-digitization.

Arjan Kuijper holds a chair in Mathematical and Applied Visual Computing at TU Darmstadt and is a member of the management of Fraunhofer IGD, responsible for scientific dissemination. He obtained his M.Sc. degree in applied mathematics from Twente University and Ph.D. degree from Utrecht University, both in the Netherlands. He was assistant research professor at the IT University of Copenhagen, Denmark, and senior researcher at RICAM in Linz, Austria. He obtained his habilitation degree from TU Graz, Austria. He is the author of over 300 peer-reviewed publications, associate editor for CVIU, PR, and TVCJ, secretary of the International Association for Pattern Recognition (IAPR) and serves both as reviewer for many journals and conferences, and as program committee member and organizer of conferences. His research interests cover all aspects of mathematics-based methods for computer vision, graphics, imaging, pattern recognition, interaction, and visualization.

Dieter W. Fellner is a professor of computer science at TU Darmstadt, Germany, and director of the Fraunhofer Institute for Computer Graphics Research IGD at the same location since Oct. 2006. He is still affiliated with the Graz University of Technology where he chairs the Institute of Computer Graphics and Knowledge Visualization he founded in 2005. Dieter W. Fellner's research activities over the last years as documented in more than 370 publications covered algorithms and software architectures to integrate modeling and rendering, efficient rendering and visualization algorithms, generative and reconstructive modeling, virtual and augmented reality, graphical aspects of internet-based multimedia information systems and cultural heritage as well as digital libraries. He is a member of EUROGRAPHICS, ACM, IEEE Computer Society, and the Gesellschaft fr Informatik (GI).